PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-296357
(43)Date of publication of application : 29.10.1999

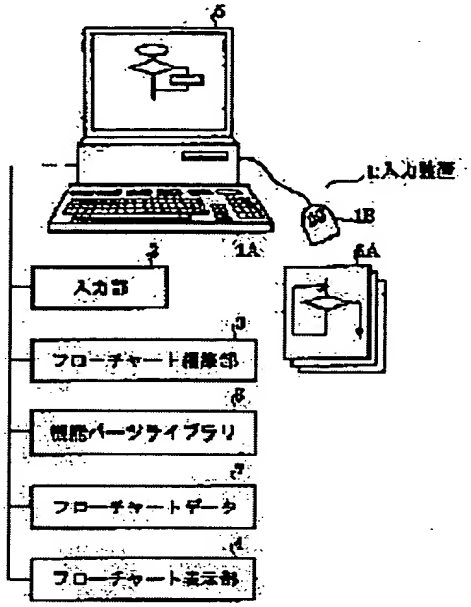
(51)Int.Cl. G06F 9/06

(21)Application number : 10-112756
(22)Date of filing : 08.04.1998

(71)Applicant : OKI ELECTRIC IND CO LTD
(72)Inventor : YAMADA ZENSHI

(54) FLOW CHART INPUT DEVICE

(57)Abstract:
PROBLEM TO BE SOLVED: To easily generate a correct flow chart without an error in PRO GRAM by arranging function parts selected by an input device to designated positions and editing flow chart data which indicates a flow chart structure.
SOLUTION: A user generates a flow chart while viewing a display device 5 through the use of the input device 1. The device 1 is constituted of the parts such as a keyboard 1A and a mouse 1B, for displaying the intention of the user in the device. In this case, the device 1 selects the optional function parts stored in a function parts library 6 and designates that they are displayed at the prescribed positions on a display device 5. Then, an input part 2 analyzes the operation of the user, judges the propriety of the operation and gives a proper indication to a flow chart editing part 3. The part 3 executes a processing for editing flow chart data 7 in accordance with the intention of the user.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平11－296357

(43)公開日　平成11年(1999)10月29日

(51)Int.Cl.[6]	識別記号	FI	
G06F 9/06	530	G06F 9/06	530H
			530W

審査請求　未請求　請求項の数13　FD　(全 18 頁)

(21)出願番号　特願平10－112756

(22)出願日　平成10年(1998)4月8日

(71)出願人　000000295
沖電気工業株式会社
東京都港区虎ノ門1丁目7番12号

(72)発明者　山田　善嗣
東京都港区虎ノ門1丁目7番12号　沖電気工業株式会社内

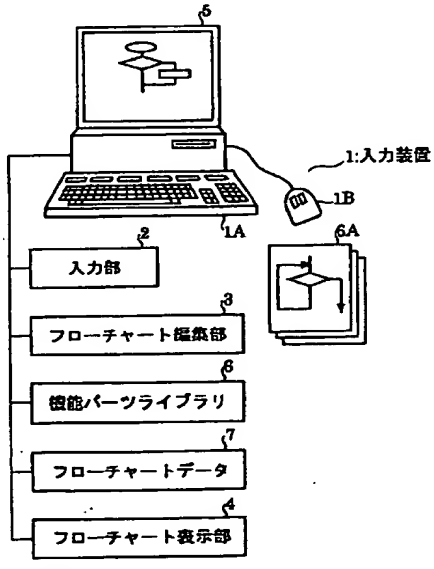
(74)代理人　弁理士　佐藤　幸男　(外1名)

(54)【発明の名称】　フローチャート入力装置

(57)【要約】

【解決手段】　表示装置５にフローチャートを描く際、機能パーツライブラリ６を使用する。機能パーツは、フローチャート中に使用されて一定の処理を完結する手順を表現する図形パーツ群により構成される。機能パーツ群を、許された場所に許された順で挿入し、フローチャートを組み立てる。

【効果】　論理的に矛盾のないイベントドリブン的なフローチャートを容易に作成でき、メカトロニクス製品のソフトウェア開発が容易になる。



本発明のフローチャート入力装置ブロック図

1

【特許請求の範囲】

【請求項１】　フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、機能パーツのイメージを複数格納した機能パーツライブラリと、

前記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定する入力装置と、

入力装置により選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するフローチャート編集部とを備えたことを特徴とするフローチャート入力装置。

【請求項２】　請求項１に記載のフローチャート入力装置において、

各機能パーツには、予め他の機能パーツを挿入して配置することができる位置を表示する空パーツが含められることを特徴とするフローチャート入力装置。

【請求項３】　請求項２に記載のフローチャート入力装置において、

入力装置により指定された機能パーツの挿入位置が、既に表示装置に表示された機能パーツの空パーツ上でないときは、挿入を拒絶する挿入判定部を備えたことを特徴とするフローチャート入力装置。

【請求項４】　請求項３に記載のフローチャート入力装置において、

各機能パーツの空パーツ部分に挿入することができる機能パーツを予め限定する挿入判定マトリクスを備え、

挿入判定部は、前記挿入判定マトリクスを参照して、入力装置により選択された機能パーツの挿入を拒絶するかどうか判定することを特徴とするフローチャート入力装置。

【請求項５】　請求項１に記載のフローチャート入力装置において、

フローチャート編集部は、

既に表示装置に表示されたフローチャート中に、新たな機能パーツが挿入されたときは、次に他の機能パーツを挿入して配置することができる位置を表示する空パーツを、前記フローチャート中の機能パーツ間に配置することを特徴とするフローチャート入力装置。

【請求項６】　請求項１に記載のフローチャート入力装置において、

各機能パーツには、予め、プログラムの流れを示す矢印を接続することができる位置を表示する接続点マークが含められることを特徴とするフローチャート入力装置。

【請求項７】　請求項６に記載のフローチャート入力装置において、

表示装置上に表示された任意の機能パーツに含まれる矢印の接続位置が、入力装置により指定されたとき、既に表示装置に表示された機能パーツの接続点マーク上でない場合にその接続を拒絶する接続判定部を備えたことを特徴とするフローチャート入力装置。

2

【請求項８】　請求項７に記載のフローチャート入力装置において、

接続判定部は、指定された矢印の接続位置が既に表示装置に表示されている機能パーツとの関係から所定の基準を満たさないときは、入力装置により指定された矢印の接続を拒絶することを特徴とするフローチャート入力装置。

【請求項９】　請求項１に記載のフローチャート入力装置において、

フローチャート編集部は、

フローチャートデータを階層構造により表現し、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層の下に、前記状態を示すステップの属性情報を含むデータ階層を設けるようにしたことを特徴とするフローチャート入力装置。

【請求項１０】　請求項９に記載のフローチャート入力装置において、

フローチャートデータ中の、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層を選択するとともに、前記状態遷移の原因となるイベントをさらに下層のデータ階層から抽出して状態遷移図を生成する状態遷移図生成部を備えたことを特徴とするフローチャート入力装置。

【請求項１１】　請求項１に記載のフローチャート入力装置において、

各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するプログラム生成部を備えたことを特徴とするフローチャート入力装置。

【請求項１２】　フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、

予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、

入力装置により、前記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、

選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【請求項１３】　フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、

予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、

入力装置により、前記機能パーツライブラリに格納され

た任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、
選択された前記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するとともに、
各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ソフトウェア開発、特にメカトロニクス製品の開発等を支援するためのドキュメントからプログラムへの自動変換可能なＣＡＳＥ（computer aidedsoftware engineering）ツールとなり得る、フローチャート入力装置に関する。

【０００２】

【従来の技術】メカトロニクス製品は、論理回路、機構、ソフトウェアの３要素からなる。そのため、機構設計者、ソフトウェア設計者、回路設計者の三者が共同して製品開発を進めなければならない。通常、機構をどのように制御し、どのように動作させるかというイメージは、その機構を設計した設計者が持っている。しかしながら、一般に機構設計者はそのイメージをソフトウェアとして実現する技術を持たない。一方、ソフトウェア設計者は、ソフトウェアを実現する技術を持っていても、新たに設計された機構の制御方法や動作は機構設計者から聞かなければならない。従って、機構設計者は、機構の制御、動作イメージをソフトウェア設計者に伝達するためのドキュメントを作成する必要がある。

【０００３】ソフトウェア設計者は機構設計者から受け取ったドキュメントをもとに、ソフトウェアを開発する。このため、そのドキュメントには矛盾や曖昧さがないこと、プログラム的に見て正しいことが要求される。しかし、機構設計者はソフトウェア設計の技術を持たないため、こうしたドキュメントを作成するのは容易でない。従って、ソフトウェア設計の技術を持たない者でも正しいドキュメントを容易に描き得るツールが必要となる。

【０００４】また、メカトロニクス製品のソフトウェアは、高度な処理速度性能が要求される。そのため、割り込みを多用したイベントドリブン指向のプログラムとなる。従って、機構作成者が作成するドキュメントも、現実のプログラムに即したイベントドリブン指向で記述されていることが望ましい。

【０００５】即ち、メカトロニクス製品の開発に適用するドキュメント作成ツールには次のような用件が望まれる。１．ソフトウェアの知識がなくても、正しいドキュメントが作成できること。２．イベントドリブン指向の記述が可能なこと。

【０００６】一方、従来から行われているドキュメントの作成方法には以下のようなものがある。

１．手作業でフローチャートを描く。

２．ドロー系ツールを使用してフローチャートを描く。

３．ＨＣＰチャート、ＰＡＤ、ＨＩＰＯ、ＮＳチャートと呼ばれるソフトウェア開発専用のＣＡＳＥツールを使用する。

４．フローチャートを採用したＣＡＳＥツールを使用する。

【０００７】

【発明が解決しようとする課題】ところで、上記のような従来の技術には次のような解決すべき課題があった。フローチャートを手作業で描くには多大な工数が必要になる。また、記述の自由度が大きすぎて、フローチャートの正当性のチェックも人に頼らざるを得ない。更に、ソフトウェアの知識がないと、システムのサポートがなければ、正しいフローチャートを作成することが困難である。フローチャート用のテンプレートを用意したドロー系のツールを使用すれば、図形描画が容易になり、入力工数は削減される。しかし、その他の問題点は手作業と変わらない。

【０００８】一方、ＨＣＰチャート、ＰＡＤ、ＨＩＰＯ、ＮＳチャート等のＣＡＳＥツールは、フローチャートを用いないため、上記の問題が解決される。また、この手法自体、フローチャートに比べて適切に自由度が制限され、比較的正しいドキュメントを書き易い。また、プログラムへの自動変換も考慮して開発がされている。しかしながら、これらの手法は、まだ一般的でなく、標準化もなされておらず、特にソフトウェア設計者でない者にとって馴染みが薄いという問題がある。

【０００９】フローチャートを使用したＣＡＳＥツールは多くは存在しない。特に、プログラムに自動変換可能なものはまれである。また、フローチャートという手法自体の問題点から、ｇｏｔｏレスの構造化されたプログラムに変換できるものは見当たらない。即ち、従来のＣＡＳＥツールはソフトウェア設計者を対象にしたものが多く、プログラミングの知識がないと正しいドキュメントを作成することは困難である。また、これらのツールには組込み機器用即ちメカトロニクス製品用に特化されたものは少ない。そのため、イベントドリブン指向のドキュメントを作成できるものが少ない。

【００１０】

【課題を解決するための手段】本発明は以上の点を解決するため次の構成を採用する。

〈構成１〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、機能パーツのイメージを複数格納した機能パーツライブラリと、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の

所定の位置に表示するように指定する入力装置と、入力装置により選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するフローチャート編集部とを備えたことを特徴とするフローチャート入力装置。

【００１１】〈構成２〉構成１に記載のフローチャート入力装置において、各機能パーツには、予め他の機能パーツを挿入して配置することができる位置を表示する空パーツが含められることを特徴とするフローチャート入力装置。

【００１２】〈構成３〉構成２に記載のフローチャート入力装置において、入力装置により指定された機能パーツの挿入位置が、既に表示装置に表示された機能パーツの空パーツ上でないときは、挿入を拒絶する挿入判定部を備えたことを特徴とするフローチャート入力装置。

【００１３】〈構成４〉構成３に記載のフローチャート入力装置において、各機能パーツの空パーツ部分に挿入することができる機能パーツを予め限定する挿入判定マトリクスを備え、挿入判定部は、上記挿入判定マトリクスを参照して、入力装置により選択された機能パーツの挿入を拒絶するかどうか判定することを特徴とするフローチャート入力装置。

【００１４】〈構成５〉構成１に記載のフローチャート入力装置において、フローチャート編集部は、既に表示装置に表示されたフローチャート中に、新たな機能パーツが挿入されたときは、次に他の機能パーツを挿入して配置することができる位置を表示する空パーツを、上記フローチャート中の機能パーツ間に配置することを特徴とするフローチャート入力装置。

【００１５】〈構成６〉構成１に記載のフローチャート入力装置において、各機能パーツには、予め、プログラムの流れを示す矢印を接続することができる位置を表示する接続点マークが含められることを特徴とするフローチャート入力装置。

【００１６】〈構成７〉構成６に記載のフローチャート入力装置において、表示装置上に表示された任意の機能パーツに含まれる矢印の接続位置が、入力装置により指定されたとき、既に表示装置に表示された機能パーツの接続点マーク上でない場合にその接続を拒絶する接続判定部を備えたことを特徴とするフローチャート入力装置。

【００１７】〈構成８〉構成７に記載のフローチャート入力装置において、接続判定部は、指定された矢印の接続位置が既に表示装置に表示されている機能パーツとの関係から所定の基準を満たさないときは、入力装置により指定された矢印の接続を拒絶することを特徴とするフローチャート入力装置。

【００１８】〈構成９〉構成１に記載のフローチャート入力装置において、フローチャート編集部は、フローチャートデータを階層構造により表現し、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層の下に、上記状態を示すステップの属性情報を含むデータ階層を設けるようにしたことを特徴とするフローチャート入力装置。

【００１９】〈構成１０〉構成９に記載のフローチャート入力装置において、フローチャートデータ中の、状態を示すステップに関する情報と、状態遷移を示す矢印に関する情報とを含むデータ階層を選択するとともに、上記状態遷移の原因となるイベントをさらに下層のデータ階層から抽出して状態遷移図を生成する状態遷移図生成部を備えたことを特徴とするフローチャート入力装置。

【００２０】〈構成１１〉構成１に記載のフローチャート入力装置において、各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するプログラム生成部を備えたことを特徴とするフローチャート入力装置。

【００２１】〈構成１２〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、入力装置により、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【００２２】〈構成１３〉フローチャート中に使用されて、一定の処理を完結する手順を表現するための図形パーツ群を機能パーツと呼ぶとき、予め、機能パーツのイメージを複数格納した機能パーツライブラリを用意して、入力装置により、上記機能パーツライブラリに格納された任意の機能パーツを選択して、表示装置上の所定の位置に表示するように指定したとき、選択された上記機能パーツを、指定された位置に配置するようにして、フローチャートの構造を示すフローチャートデータを編集するとともに、各機能パーツに対応させて用意されたプログラムソースコードと、フローチャートデータとを編集して、フローチャートを実行するためのソースプログラムを自動生成するように動作する、フローチャート入力用のコンピュータプログラムを記録した記録媒体。

【００２３】

【発明の実施の形態】以下、本発明の実施の形態を具体例を用いて説明する。

〈具体例１〉図１は、本発明のフローチャート入力装置を示すブロック図である。図に示すようにこの装置は、入力装置１、入力部２、フローチャート編集部３、フローチャート表示部４、表示装置５、機能パーツライブラリ６、フローチャートデータ７等から構成される。

【００２４】ユーザは、入力装置１を用いて表示装置５を見ながらフローチャートを作成する。入力装置１は、キーボード１Ａやマウス１Ｂ等、ユーザの意志を装置に指示するための部品により構成される。入力部２は、ユーザの操作を解釈し、その操作の可否を判断し、フローチャート編集部３に適切な指示を出す処理を行う部分である。フローチャート編集部３は、ユーザの意志に従ってフローチャートデータ７を編集する処理を行う部分である。

【００２５】フローチャート表示部４は、フローチャートデータ７の更新状態を監視して、編集されて更新されたフローチャートデータをそのつど表示装置５に表示する制御を行う部分である。更に、フローチャート表示部４は、ユーザに対し操作の可否を知らせる処理も行う。表示装置５は、ブラウン管ディスプレイや液晶表示盤等から成り、ユーザに編集結果を知らせるための装置である。機能パーツライブラリ６は、後で詳しく説明するように、ユーザがフローチャートを構築する上でその部品となるイメージや情報を格納する部分である。フローチャートデータ７は、フローチャート編集部３によって編集された、後で説明するような構造のデータから構成される。

【００２６】以上の装置は、例えばワークステーションやパーソナルコンピュータに上記の機能ブロックを実現するコンピュータプログラムをインストールして実現される。入力部２、フローチャート編集部３、フローチャート表示部４等は、それぞれの機能を実現するプログラムモジュールにより構成される。また、機能パーツライブラリ６やフローチャートデータ７は、コンピュータ中に設けられた記憶装置の内部に蓄積された所定の構成のデータにより実現される。

【００２７】図２に、図１に示した表示装置５に表示されたウインドウの概略図を示す。この画面は、フローチャート入力作業を行う場合に表示される画面で、メインウインドウ１１とパーツパレット１２とから構成される。メインウインドウ１１には、フローチャートを作成し表示するためのフローチャート表示エリア１３と、ユーザが各種の操作を指示するためのメインメニュー１４と、マウスを用いてクリックし具体的な操作を直接指示するツールボタン群１５を備えている。パーツパレット１２には、図１に示した機能パーツライブラリ６の様々な機能パーツ６Ａを選択するボタンが並んでいる。なお、ツールボタン群１５や機能パーツ６Ａを選択するボタンの細部の図示は、ここでは省略している。

【００２８】従来は、パーツパレットにより菱形や四角形、矢印といった図形パーツのテンプレートを用意し、こうしたフローチャートを構築するための部品として使用していた。しかしながら、図形パーツ自体はプログラムとして意味を持たず、フローチャート作成上の自由度が大きすぎる。そこで、本発明では次のような機能パーツを単位としてフローチャートの作成を補助する。即ち、図２のパーツパレット１２は、機能パーツを１単位の部品としている。

【００２９】図３に、機能パーツの具体例説明図を示す。図３の（ａ）～（ｆ）には、本発明でフローチャート作成に使用する機能パーツの様々な例を示した。機能パーツは、それぞれ、プログラム中に使用されて一定の処理を完結する手順を表現するための図形パーツ群から構成される。図形パーツというのは、既に説明したように、菱形や四角形、矢印等のことである。

【００３０】例えば、図３（ａ）は、「ステップ」と呼ばれる機能パーツである。これは、待機ループ中でイベントを発生すると、別の状態に遷移する機能を持ち、待機ループ１６中に判断用の菱形１７を配置したものである。これは状態遷移図の１つの状態に相当する。また、状態遷移マトリクス表の列に相当する。このパーツが通常のループと異なるのは、通常のループが処理の繰り返しであるのに対し、ステップの機能パーツはタスクのウエイト状態に移行する点である。

【００３１】イベント発生のチェックはオペレーティングシステムに任される。オペレーティングシステムはイベントの発生を検知すると、そのイベントに対応した処理を実行する。イベントドリブン指向のプログラムとは、こうしたステップの組み合わせで記述されたプログラムを指している。なお、同一の状態で複数種類のイベント待ちを行うことがあるから、待機ループ１６中に後から任意の数の菱形１７を挿入することができる。

【００３２】図３（ｂ）に示す分岐という機能パーツは、菱形１７の部分で一定の条件判断をし、２種類のルートに分岐する機能を持つ。各分岐後のルートには必要に応じて後から任意の処理が挿入される。これも入口から出口までの間に一定の完結した処理を実行する機能を持つ。（ｃ）は、ケース文の機能パーツである。菱形１７の部分で条件判断をし、ケースに応じて３つ以上のルートに分岐する。各ルートでは、それぞれケースに応じた処理が実行される。長方形１８は、その処理を示す。

【００３３】図３（ｄ）は、端子の機能パーツである。プログラムの開始時や終了時に、このパーツ１９が使用される。（ｅ）は前判定ループ、（ｆ）は後判定ループである。前判定ループは、菱形１７で一定の判断を行い、例えば条件を満たさない場合には長方形１８を含む処理ループを繰り返し、条件を満たした場合にはこの機能パーツの処理から抜ける。後判定ループの場合、予め長方形１８の処理を実行し、菱形１７で条件判断を行う。条件を満たさない場合には、長方形１８を含む処理ループを繰り返す。条件を満たした場合にはこの機能パーツの処理を抜ける。

【００３４】以上のように、いずれの機能パーツもプログラム的に正しい意味を持ち、一定の処理を完結し、フローチャート構築の単位となる。ユーザは、こうした機

能パーツを配列していくことによってフローチャートを組み立てるので、プログラム的に正しいフローチャートを記述することができる。また、これらの機能パーツによる処理の内容はそれぞれプログラムソースコードに対応させることができる。

【００３５】例えば（ａ）は、イベントをそのままオペレーティングシステムに伝えるメッセージ群により記述できる。（ｂ）はいわゆるｉｆ～Ｔｈｅｎ，Ｅｌｓｅ文により記述される。また、（ｃ）はｃａｓｅ文により記述される。（ｄ）の端子はｅｎｄ文である。（ｅ）や（ｆ）は、ｉｆ文やｆｏｒ文により記述される。

【００３６】図４には、具体的なソフトウェア設計者がフローチャートを設計する場合の設計例説明図を示す。例えば、図４（ａ）に示すような機構についてフローチャートを作成するものとする。図の（ａ）に示す機構は、搬送モータ２１を用いてコンベアー２２を駆動し、工作物２３を図の右側から左側の方向に搬送するように動作する。ここで、工作物２３は、コンベアー２２上を搬送される途中、第１センサ２４や第２センサ２５によって位置検出される。第１センサ２４が工作物２３の通過を検出すると一定の処理が実行される。更に、第２センサ２５が工作物２３を検出すると、別の処理が実行される。このような場合、図の（ｂ）に示すようなフローチャートを作成する。

【００３７】即ち、図４（ｂ）の処理Ｓ１において、モータがスタートすると、処理Ｓ２において、第１センサ２４が工作物２３の通過というイベント待ち状態に入る。第１センサが工作物２３を検出すると、Ｓ３の処理１に進む。そして、その後処理Ｓ４に進み、第２センサが工作物２３の通過待ちというイベント待ち状態に移る。第２センサが工作物２３の通過を検出すると、Ｓ５に進んで処理２が実行される。

【００３８】実際のメカトロニクス製品では、様々な複雑な制御が行われるが、例えば図４（ｂ）のような制御を行う場合に、第１センサの監視や第２センサの監視には、図３（ａ）を用いて説明したステップの機能パーツをそのまま利用する。後は、図３には例示しなかったが、長方形のみの処理を示す機能パーツを組み合わせる。本発明の装置は、こうしてフローチャートを作成する場合に活用される。

【００３９】図５に、フローチャート作成とプログラムへの変換例説明図を示す。この図を用いて、図１に示す装置の具体的なフローチャート入力操作を説明する。まず、ユーザが図２に示すような画面を図１に示す表示装置５に表示させる。そして、パーツパレット１２から様々な機能パーツをドラッグしてフローチャート表示エリア１３上にドロップする。この操作を繰り返すことによって、例えば図５（ａ）に示すようなフローチャートが描かれる。このフローチャートは、初期処理の機能パーツの次に図３（ａ）に示したステップの機能パーツを接続し、予めステップの機能パーツに含まれているイベント１に更にイベント２の機能パーツを追加する。その後、イベント処理１の機能パーツとイベント処理２の機能パーツを接続する。

【００４０】図１に示すマウス１Ｂの操作によって、このような機能パーツがドラッグされると、入力部２が機能パーツのドラッグ開始を検出する。ユーザは、フローチャート表示エリア１３上の所定の位置を指定して、機能パーツをドロップする。入力部２は、機能パーツのドロップを検出する。そして、そのドロップ位置がフローチャート中のどの位置かを判断する。フローチャート中の適切な位置にその機能パーツが挿入された場合、これをフローチャート編集部３に通知する。

【００４１】フローチャート編集部３は、機能パーツライブラリ６から該当する機能パーツのデータを取り出して、フローチャートデータ７の該当する部分に対応するデータを書き込む。フローチャート表示部４は、フローチャートデータ７が更新されたことを検出すると、表示装置５に対し該当する機能パーツのイメージを取り出して表示する。以上のようにして、フローチャートの入力編集が進められる。

【００４２】図５（ｂ）は、従来のドロー系のツールを使用する場合のフローチャート用図形パーツである。この図に示す菱形１７は、これだけではプログラム的に意味をなさない。図の（ｃ）に示すように、条件判断の結果分岐するルートを示す矢印等が含まれて初めて、一定の処理を完結する手順が表現される。このような機能パーツ単位でフローチャートの作成を支援することによりフローチャートの作成が容易になり、かつ、正確になる。更に、図５（ｃ）に示すように、この機能パーツはそのままｉｆ文に置き換えることができ、ソースプログラムの自動作成が可能になる。

【００４３】〈具体例１の効果〉以上のように、プログラム作成を機能パーツ単位で行い、機能パーツのイメージを格納した機能パーツライブラリと、そのフローチャート中の位置を指定する入力装置と、フローチャートデータを編集するフローチャート編集部等によってプログラム的に誤りのない正しいフローチャートが容易に作成できるという効果がある。

【００４４】〈具体例２〉上記のようにフローチャートを作成していく場合に、機能パーツを相互に接続しあるいは機能パーツの各部に別の機能パーツを挿入する作業が進められる。ここでは、ユーザによる機能パーツの挿入位置が正しいかどうかを判断し、フローチャートの作成を支援する例を説明する。

【００４５】図６には、これを実現するための装置ブロック図を示す。この図は、図１に示した装置の構成を基準とし、同一部分には同一符号を付している。この具体例２の装置には、図１に示した装置に挿入判定部３１を追加している。

【００４６】図７には、挿入判定用表示例説明図（その１）を示す。図の（ａ）は、入口端子の機能パーツと出口端子の機能パーツを組み合わせたものである。実際には、この間に様々な機能パーツが挿入されてフローチャートが完成する。図に示す入口端子と出口端子の中間に○印を表示している。この○印は、予め他の機能パーツを挿入して配置することができる位置を表示するためのもので、本発明では空パーツと呼ぶことにする。

【００４７】各機能パーツには、予めこのような空パーツを含めておき、パーツパレットからドラッグしてきた機能パーツをドロップすることができるポイントをガイドする。なお、この○印で示した空パーツと重ねて表示されている×印は、後で説明するフローチャート中の矢印の先を接続することができる接続点マークを表す。

【００４８】図の（ｂ）には、（ａ）のフローチャート中にいくつかの機能パーツが挿入された新たなフローチャートを示す。図中の、例えば破線の○印は、イベント２を含むステップ中に別のイベントを挿入することができる空パーツである。また、◎の空パーツは、各機能パーツの入り口や出口に配置されている。○印の空パーツは、各機能パーツの間に配置されている。これらの、○印や◎印、破線の○印といった空パーツは、必要に応じてそれぞれ挿入することができる機能パーツの種類を区別するために色分けされたり、形状を変えて表示されたものである。

【００４９】そして、ユーザがパーツをドラッグしている間、マウスカーソルは、ドロップ可能な位置にいるときと、不可能な位置にいるときで、その形状を変化させる。こうして、ユーザに機能パーツのドロップ位置を意識させることができる。また、ドロップ不可能な位置にユーザが機能パーツをドロップしようとしても、システムはその操作を無視する。こうした動作自体は、既存のソフトウエアにより既に広く実施されており、容易に実現できる。

【００５０】次に、本発明の装置の動作を説明する。まず、ユーザは入力装置１を使って、図２に示したようなパーツパレット１２中の機能パーツをドラッグする。入力部２は、機能パーツのドラッグ開始を検出する。入力部２は、ドラッグ中のパーツの位置を順次検出して、その位置情報を挿入判定部３１に渡す。挿入判定部３１は、ドラッグ中のパーツ位置が空パーツの上にあるかどうかを判定する。そして、その結果をフローチャート表示部４に通知する。フローチャート表示部４は、入力部２の指示に従ってドラッグ中のマウスカーソルの形状を変更して、ユーザにその位置がドロップ可能な位置かどうかを通知する。

【００５１】機能パーツがドロップされた場合、入力部２はそれを検知し、ドロップされた位置を挿入判定部３１に通知する。挿入判定部３１は、ドロップされた位置が空パーツ上かどうかを判断し、不適切であればその操作を無視する。また、挿入判定部３１は、ドロップされた位置が空パーツ上であれば、フローチャート上の該当位置にパーツが挿入されたことをフローチャート編集部３に通知する。

【００５２】フローチャート編集部３は、機能パーツライブラリ６から機能パーツのデータを取り出し、フローチャートデータ７の該当する部分にパーツデータを挿入する。フローチャート編集部３は、新たな機能パーツが挿入されたことによって、新たに必要となる空パーツをフローチャートデータ７に挿入する。フローチャート表示部４は、フローチャートデータ７が更新されたことを検出すると、その結果を表示装置５に出力しフローチャートを更新して表示する。

【００５３】従来、例えばドロー系のツールを用いてフローチャートを作成した場合には、図形パーツを自由に配置することができ、それらのパーツに正しい前後関係を与えるのはユーザの責任に任されていた。従って、ユーザのレベルによって不適切なパーツ配置を行うことも少なくなかった。

【００５４】この具体例では、常に適切な位置にのみパーツが挿入されるように挿入可能な箇所を指定し、これを挿入判定部３１において判定する。従って、例えば挿入した機能パーツがどこにもつながらずに浮いてしまうといった誤りが防止でき、常にプログラム的に正しいフローチャートの入力ができる。更に、機能パーツを挿入したポイントにより、挿入された機能パーツがプログラム上のどの部分に挿入されたかをも認識することが可能になる。この動作を具体的に説明する。

【００５５】図８には、挿入判定用表示例説明図（その２）を示す。図の（ａ）に示すように、このフローチャートがステップ１という機能パーツとステップ２という機能パーツとを接続して構成されているものとする。このとき、一般には、図中の（１）という部分に挿入した処理は、ステップ１のイベント処理なのか、ステップ２を実行する前の初期処理なのかを判別することができない。

【００５６】ところが、既に説明したように、例えば各機能パーツの入口と出口に◎の空パーツを配置しておく。そして、（ｂ）に示すように、ステップ１とステップ２の２組の機能パーツを接続した場合、その接続点に新たな○印の空パーツを加える。（２）の部分にある◎印の空パーツはステップ１に属する処理である。（３）の部分にある◎印の空パーツはステップ２に属する処理である。そして、○印の空パーツは新たな機能パーツを挿入することができる部分である。これを利用することによって、この部分に新たに挿入された機能パーツがステップ１あるいはステップ２に属する処理なのか、独立した処理なのかを区別することができる。

【００５７】従って、各機能パーツを、既に説明した要領でソースコードに置き換え、ソースプログラムを生成

する場合に、新たに挿入された機能パーツのソースコードをどの機能パーツのソースコードに含めるかを判断することが可能になる。もちろん、周辺の空パーツを表示装置５に表示し続ければ、フローチャート作成者にも、その区別ができる。

【００５８】〈具体例２の効果〉以上のように、フローチャート中に機能パーツを挿入することができる場所を限定し、挿入判定部３１によってこれを判定するように制御すれば、誤った場所に機能パーツを挿入するといったミスを防止できる。また、プログラムの流れを明確にし、機能パーツ単位のソースプログラムへの自動変換が容易になる。

【００５９】〈具体例３〉具体例２を用いて説明したように、各機能パーツに予め他の機能パーツを挿入できる位置を空パーツとして設定しておけば、フローチャートの作成が容易になるが、挿入可能な機能パーツを制限すれば、より誤りの無いフローチャートの入力が可能になる。そこで、挿入可能な機能パーツを判定するためのデータを設定する。

【００６０】図９には、具体例３の装置ブロック図を示す。この装置は、具体例２の装置に対し、挿入判定マトリクスを新たに追加したものである。その他の部分は具体例２の装置と同一の構成であり、同一の符号を用いて表現した。

【００６１】図１０（ａ）には、挿入判定マトリクスの内容説明図を示す。また、（ｂ）には、挿入判定マトリクスを参照して動作する挿入判定部の動作説明図を示す。図１０の（ａ）に示すように、左側に示した空パーツは、それぞれ○、◎、破線の○及び太線の○というように区別されている。これらの空パーツがフローチャート中に存在する場合には、それぞれ図の最上段に示した様々な機能パーツ例えばステップ、分岐、イベント処理、端子、前判定、後判定、ケース、結合子等の機能パーツのうち、○が付いたものだけをそこに挿入することができる。

【００６２】図の（ａ）に示す例では、例えば破線の○印はイベント以外の挿入を禁止している。この破線の○印は、図７（ｂ）に示すようなステップの機能パーツ中に表示されていた。ステップは、イベント待ちの状態を示すもので、一般的なループ処理とは異なる。従って、ステップ中にはイベント以外を挿入することができない。ユーザは、このような空パーツの形状や種類によって、ドラッグしている機能パーツの挿入可能な位置を見分けることができる。具体的には次のようにして装置が動作する。

【００６３】まず、機能パーツがドロップされた場合、図９に示す入力部２はそれを検知し、ドロップされたパーツの種類とドロップされた位置情報を挿入判定部３１に通知する。挿入判定部３１は、ドロップされたパーツと空パーツを示す○印の種類が挿入可能な組み合わせかどうかを判定する。これは、図１０（ａ）に示したような挿入判定マトリクス３２を参照して行う。

【００６４】挿入判定マトリクス３２で、妥当な組み合わせであるという表示がされていない場合には、ドロップ操作を無視する。挿入判定部３１が判定した結果、妥当な組み合わせであれば、フローチャート上の該当位置に機能パーツが挿入されたことをフローチャート編集部３に通知する。フローチャート編集部３は、機能パーツライブラリ６から該当する機能パーツのデータを取り出し、フローチャートデータ７の該当位置に挿入する。そして、フローチャート編集部３は、パーツが挿入されたことによって新たに必要になる空パーツをフローチャートデータ７に挿入する。フローチャート表示部４は、その後フローチャートデータ７の更新を検出し、表示装置５に新たなフローチャートを表示する。

【００６５】例えば、図１０（ｂ）において、破線で囲まれたステップの機能パーツが既に描かれており、イベント１とイベント２のパーツが挿入されていたとする。これに新たに処理のパーツを挿入すると、これはイベント待ちのステップとして意味をなさなくなる。ループの中に処理が挿入されることによって、タスクのウエイト状態への移行は不可能になってしまい、通常の繰り返し処理と同じになるからである。

【００６６】更に、本来、ステップがあるべき場所にそのようなパーツが存在することによって、状態遷移図への変換も不可能になる。また、そのような不適切なパーツの挿入は、イベントドリブン指向としての正当性を損なうことになる。従って、図１０（ｂ）に示すようなフローチャートが作成できないようにすることによって、この発明の目的は達成される。

【００６７】〈具体例３の効果〉以上に示すように、挿入判定マトリクスを設けることによって、空パーツの部分に挿入可能な機能パーツの種類を限定することができる。こうして不適切なパーツの挿入によって誤ったフローチャートの作成を防止できる。

【００６８】〈具体例４〉具体例３では、機能パーツを挿入するべき位置を判断した。プログラムのフローチャート入力作業では、機能パーツを挿入することの他に、イベント処理をした後の状態遷移先を矢印の接続がある。この矢印の接続に誤りがあれば論理的な矛盾が生じ、更にイベントドリブン指向としてのプログラムの正当性を損なう。

【００６９】図１１には、具体例４の装置のブロック図を示した。この装置には、矢印接続の正当性を判断する接続判定部３３を設けた。この他の機能ブロックは、具体例３を用いて説明したものと同様であり、それぞれ同一の記号を付している。

【００７０】図１２は、本発明によって描かれたフローチャートの一例である。この図を用いて、接続判定動作の説明を行う。この図の右側に示したＴｈｅｎ処理と、

Ｅｌｓｅ処理に続く部分には、（４）あるいは（５）に示すような矢印が設けられている。矢印の首部にある黒塗りの四角形をカーソルでクリックしこれをドラッグすることによって、矢印を任意の場所に移動できる。そして、指定した場所にドロップすると、その矢印の先端がフローチャート中に接続される。ここで、矢印の先端が接続可能な場所に×印の接続点マークを示した。

【００７１】即ち、図の（１）、（２）、（３）以外の場所にはこれらの矢印の先端を接続することができない。即ち、プログラムのイベントドリブン的な性格を保持するために、矢印は、状態遷移先として適切な場所にのみ接続することができる。なお、図の（６）に示した矢印はステップの機能パーツに含まれる。この矢印の先を移動させると機能パーツ自身の正当な動作を妨げる。従って黒塗りの四角形が用意されておらず、その移動は認められない。図３（ｅ）や（ｆ）に示した前判定ループ、（ｆ）に示した後判定ループの矢印も同様に接続変更ができない。

【００７２】例えば、状態遷移以外の目的で矢印の先端を接続すると、イベントドリブン的な処理が不可能になる。このために以上のような制限が加えられる。そして、フローチャート入力中に許可されない接続を行おうとした場合には、矢印の移動は認められず元の場所に戻る。

【００７３】図１２に示す例では、（１）を起点とするステップの機能パーツは、２種のイベントに従って状態遷移する。上側に配置されたイベント１により状態遷移をする場合、図の右側に示す条件分岐の機能パーツによる処理に進む。条件を満たせばＴｈｅｎ処理、条件を満たさない場合にはＥｌｓｅ処理が実行される。

【００７４】一方、（１）で始まるステップの下側のイベント２で状態遷移する場合、（２）の直上にあるイベント処理が実行されて、（２）の部分から始まるステップに進む。こうして、次のイベント３を待つ状態に遷移する。このため、（４）や（５）で示す矢印は、状態遷移の境界である（１）、（２）あるいは（３）以外の場所には移動できない。こうした基準によって、上記接続判定部３３は動作する。

【００７５】図１３には、図１２の（５）に示した矢印を（３）の位置まで移動した後の接続判定動作の説明図（その２）を示す。この図を用いて、図１１に示した装置の動作を説明する。まず、ユーザは入力装置１を使って、フローチャート表示エリア１３上に表示された（５）の矢印先端をクリックする。そして、そのままドラッグする。入力部２は、入力装置１で矢印のドラッグが開始されたことを検知すると、接続判定部３３にその旨を接続する。接続判定部３３は、ドラッグされた矢印がドラッグ可能かどうかを判定し、（６）のようにドラッグが禁止された矢印であればその操作を無視する。

【００７６】また、入力部２は、入力装置１からドラッグ中の矢印先端位置を順次検出し、その位置情報を接続判定部３３に通知する。接続判定部３３は、受け取った矢印の位置情報が接続可能な×印上にあるかどうかを判定する。そして、その結果をフローチャート表示部４に通知する。フローチャート表示部４は、入力部２の指示に従ってドラッグ中のマウスカーソルの形状を変更し、ユーザに対しその位置への接続可否を通知する。

【００７７】矢印先端がドロップされた場合、入力部２はそれを検出し、ドロップされた位置を接続判定部３３に通知する。接続判定部３３は、ドロップされた位置が適切な×印上かどうかを判断し、不適切であればその矢印をデフォルトの位置即ちドラッグ開始前の位置に再接続するよう、フローチャート編集部３に通知する。接続判定部３３は、ドロップされた位置が適切な×印上であると判断したとき、その矢印を指定された位置に接続するようにフローチャート編集部３に指示する。

【００７８】フローチャート編集部３は、フローチャートデータ７の矢印接続先を指示された位置になるように変更する。その後、フローチャート表示部４はフローチャートデータ７が更新されたことを検出して、表示装置５にフローチャートの再表示を指示する。

【００７９】図１４には、誤ったフローチャートの例を示す。この図に示すフローチャートは、条件分岐を行った結果の分岐先を示す矢印が前判定ループの中に飛び込んでいる。こうしたフローチャートは、ｇｏｔｏ文を使わざるを得ない。ｇｏｔｏ文の使用は、プログラムの可読性、保守性を損ねるとして、一般的にその使用を避けることが望ましいといわれている。

【００８０】即ち、こうした流れは、状態遷移を中心としたイベントドリブン式構造化プログラミングの思想に反するため不適切である。従来のツールを用いた場合には、こうしたフローチャートを描くことは自由であり、これを禁止するのはユーザの意志に任されていた。この発明では、予め矢印を接続する場所が制限されて、接続判定部３３がそれを判断するから、こうした不適切な矢印の接続を禁止できる。

【００８１】〈具体例４の効果〉以上のように、この発明では、フローチャート中への不適切な矢印の接続を禁止でき、例えばユーザのプログラミング能力のレベルによらず、常に構造化されたｇｏｔｏレスのフローチャートを作成するようにユーザ支援ができる。また、論理的に誤りのある接続を防止することもでき、その後のバグのないプログラムの自動生成を可能にする。

【００８２】従って、たとえ上記のような構造化プログラミングによる制限に反しなくても、その矢印の接続を行った場合に、不適切な動作が発生するような全ての場合に、矢印の接続を禁止する処理に広く利用できる。

【００８３】〈具体例５〉上記のような本発明の装置において、フローチャートデータは、機能パーツの組み合わせにより表現できる。従って、例えば機能パーツの名

称と相互の接続関係を明らかにすれば、フローチャートデータを生成することができる。また、こうしたフローチャートがあれば、各機能パーツを対応するソースコードに置き換え、機能パーツと状態遷移をする先を示す矢印をソースコード化することによって、自動的なソースプログラム生成が可能となる。

【００８４】しかしながら、表示装置に機能パーツを表示する場合には、機能パーツを構成する各図形部品のイメージを適切にリンクさせ、変形に対応する必要がある。こうした場合に、各図形パーツの効率的な的確な管理が必要となる。そこで、この具体例では、以下に説明するような階層構造のフローチャートデータを用いる。

【００８５】図１５に、具体例５の装置ブロック図を示す。ここに示すブロック図は、具体例４の装置に対し、階層構造フローチャートデータ３４を採用した点が異なる。その他の部分は具体例４と同一符号を付した。

【００８６】次の図１６、図１７、図１８を用いて、階層構造フローチャートデータの構成を説明する。図１６は、ステップ１とステップ２とステップＳ３の状態を持つプログラムのフローチャートを示す。このフローチャートには、ステップ１から直接ステップ３へ状態遷移する動作とステップ１からステップ２を経てステップ３へ状態遷移をするプログラムが示されている。ここで、ステップ１からステップ２へ状態遷移する過程で実行されるイベント処理の直前の（１）の部分に、新たな機能パーツを挿入することを考える。

【００８７】図１７には、図１６中の（１）の部分に分岐条件処理を行う機能パーツを追加したところを示す。図１６に示すフローチャートでは、ステップ１の状態からイベント１－１が発生した場合に、対応するイベント処理を行ってステップ２に進む。また、イベント１－２が発生した場合には、対応するイベント処理を実行して、ステップ３に進む。ステップ２の状態では、イベント２－１が発生した場合に対応するイベント処理を実行し、ステップ３に進む。ステップ３は、イベント３－１が発生した場合に、このプログラムを終了する処理を行う。図１７では、イベント１－１が発生した場合に、対応するイベント処理を実行する前に、新たな機能パーツにより分岐条件判断を行う。フローチャート入力作業中に、こうした機能パーツの追加やその後の削除を行うとき、その表示制御に適する構造のフローチャートデータが存在することが望ましい。

【００８８】図１８には、図１６のフローチャートを階層構造のフローチャートデータで示した例を図示した。まず、このフローチャートは、スタートの端子とストップの端子との間に子パーツが挟まれた基本構造をしている。これがＮｏ．１のデータ階層に表示されている。Ｎｏ．１のデータ階層に示した子パーツは、図１６に示すように、ステップ１に示した機能ステップと、ステップ２に示した機能ステップと、ステップ３に示した機能ステップにより構成される。これらのステップを破線の枠内に図示した。

【００８９】そして、それぞれのステップ毎に、状態遷移先を矢印で示している。図中、赤丸と表示したのは、各ステップの入口や出口に配置した空パーツである。イベントドリブン形式のプログラムでは、基本的にこれだけで全ての処理が表現できる。上の階層と下の階層との関係は、後者が前者の属性データを示すという関係である。矢印は、ポインタデータである。例えば、スタートの端子とストップの端子の属性データは、Ｎｏ．２の階層部分に表示される。即ち、いずれも楕円とフローチャートの一部となる線とによって構成されるという内容になる。

【００９０】次に、Ｎｏ．３の階層について説明を行う。ここでは、ステップ１の属性データについてのみその例を示し、その他については同様であるから図示を省略した。ステップ１の機能ステップには、図１６に示すように、スタートの端子に接続される直線と、先頭の破線の円で示した空パーツと、イベント１－１、空パーツ、イベント１－２、空パーツといった図形パーツが存在する。さらに、イベント待ち状態を表現するループ線が存在する。

【００９１】そこで、図１８に示すＮｏ．３の階層には、先頭の線と、入口にある◎の空パーツに対応する子パーツと、イベント１－１へ進む線と、イベント１やイベント２を含めた子パーツと、ループ線という属性データが表示されている。◎の空パーツを示す子パーツは、Ｎｏ．４の階層において、緑の丸印というようにその属性が表示されている。また、Ｎｏ．４の階層に３つの青の○印と、これらに挟まれたイベント１－１やイベント１－２が、Ｎｏ．３の子パーツを示す属性データとなっている。

【００９２】また、イベント１－１やイベント１－２は、それぞれ菱形の図形と、分岐線と、更にその分岐線の先に連なる子パーツとから構成されている。これをＮｏ．５の階層に示した。そして、図１６に示す（１）のイベント処理は、緑の○印で示される空パーツの間に処理の機能パーツが挟まれた構成となっている。これをＮｏ．６の階層に示した。また、処理の機能パーツは四角の図形から構成されるから、これをＮｏ．７の階層に示した。

【００９３】以上のように、このフローチャートデータは、状態を示すステップの情報と、これらのステップ間の状態遷移を示す矢印の情報を基本となるデータ階層に表現し、各状態を示すステップ毎に、そのステップの属性データを順次階層構造で表現している。このような構成にした場合に、図１６に示す（１）の部分に図１７に示すような（２）の機能パーツを挿入したとき、図１９に示すように、フローチャートデータが書き換えられる。

【００９４】即ち、図１９には、図１８に示したＮｏ．６の階層中に、新たにＮｏ．５の階層で示した子パーツに属する「分岐」と「緑の○で表す空パーツ」を追加している。そして、「分岐」の属性を表すために、Ｎｏ．７の階層には、その形状が菱形であって、線と２つの子パーツと分岐線とにより構成されることが示されている。また、２つの子パーツは、別の新たな機能パーツを挿入できる緑の○印で示した空パーツであると表現している。

【００９５】このように、新たな機能パーツを追加する場合には、階層構造になった機能パーツのデータを、階層構造になったフローチャートの該当箇所へ一括して挿入すればよい。逆に、こうした機能パーツを外す場合には、図１９中の破線で示した階層構造の機能パーツ要素を一括して削除すればよい。即ち、「分岐」という機能パーツに必要な全ての部品が階層構造となり、フローチャートも階層構造になっているので、機能パーツを追加したり削除をしたりする場合に、一部の図形パーツが取り残されてしまったりする矛盾が発生しない。

【００９６】〈具体例５の効果〉以上説明したように、フローチャートデータを階層構造で表現すれば、機能パーツ単位でフローチャートを作成処理する作業の際に、機能パーツを追加したり削除したりして試行錯誤する作業の際、フローチャートデータの取り扱いを容易にし、誤りの発生を防止できる。従って、フローチャートに矛盾を発生させたり、意味のないパーツを残したりするという問題がない。しかも、パーツを削除した際、自動的にフローチャート中の該当部分の前後を接続することが容易になる。

【００９７】〈具体例６〉図１８に示したフローチャートデータのＮｏ．２の階層に着目する。この階層には、図に示したように、状態を示すステップの情報とこれらのステップ間の状態遷移を表す矢印の情報のみが現れている。こうしたデータを利用すれば、プログラムの基本的な構造解析を行うことができる。

【００９８】図２０は、この構造解析のための機能ブロックを図示した。図１に示した装置にこうした機能ブロックを追加することによって、フローチャートデータから、例えば図２１に示すような状態遷移図を生成し表示することができる。即ち、図２０に示す装置は、階層構造フローチャートデータ３４と、変換部３６と、状態遷移図データ３７、データ表示部３８及び表示装置５から構成される。さらに、この図には、階層構造フローチャートデータ３４や機能パーツライブラリ６を参照して、フローチャートに対応するソースプログラムを生成するプログラム生成部を図示した。

【００９９】プログラム生成部４０は、まず、上記階層構造フローチャートデータあるいはその他の構造のフローチャートデータを参照して、各状態毎に、イベントの内容とその属性データを生成する。次に、機能パーツライブラリ６を参照して、フローチャート中に使用されている機能パーツの既に説明したようなソースコードを取得する。そして、上記イベントや状態の属性データに付随したプログラムコード記述部分にそのソースコードをコピーする。これで、プログラムの自動生成ができる。これ以外の具体的なプログラム生成手順は、従来のＣＡＳＥツールで実行されている処理と同様である。

【０１００】階層構造フローチャートデータ３４は、図１８に示すような内容のデータである。変換部３６は、例えばそのＮｏ．２のデータ階層を参照して、そのデータを変換処理し、状態遷移図データ３７を生成する部分である。データ表示部３８は、この状態遷移図データ３７を表示装置５に表示する部分である。ソフト技術者がプログラム全体の動作を解析する場合、一般にその状態遷移図を作成する。これを自動的に作成し、表示装置に表示することができれば、フローチャート作成後の結果をユーザが点検し、あるいは他のソフトウェア技術者が内容を理解するのに役立つ。

【０１０１】図２１は、状態遷移図の例説明図である。この図を参照しながら、図２０に示した装置の動作を説明する。まず、始めに変換部３６は、フローチャートデータ３４を解析し、図１８に示したＮｏ．２のデータ階層から状態を示すステップを抽出する。また、変換部３６は、図１８に示すフローチャートデータを解析し、各ステップの状態を表示装置に表示する際の位置やシンボル図形等を決定する。図２１中の円内に「ステップ１」、「ステップ２」、「ステップ３」と書いたものがそのシンボル図形である。

【０１０２】次に、変換部３６は、図１８に示すＮｏ．２のデータ階層から矢印に相当するデータを抽出し、各状態を表示する、これには、例えば図２１に示す円の間を矢印のイメージでつなぐ。更に、図１８に示すＮｏ．３以下のデータ階層のイベントを抽出し、状態遷移を表す矢印にそれぞれその意味を示す情報を付加する。同時に、図１８に示したＮｏ．１の階層を参照し、スタートやストップの端子に相当する情報を付け加える。このようにして、図２１に示すようなイメージを生成し、データ表示部３８の制御によって表示装置５にその図形イメージを表示する。

【０１０３】〈具体例６の効果〉以上説明したように、フローチャートデータを上記のような構成にすることによって、状態遷移を表すデータ階層から必要な状態を抽出し、該当するイベントから状態遷移の内容を抽出して、自動的に状態遷移図を生成することができる。これによって、フローチャートがメカトロニクス製品に実装されているプログラムに属したイベントドリブン型のプログラムであるかどうかの検証を即座に行うことができ、フローチャートの作成やその解析処理に有効に活用される。

【０１０４】なお、上記の具体例１から具体例６に示し

た各発明は、初めに説明したように、それぞれパーソナルコンピュータやワークステーション等にインストールしたプログラムによって実行することができる。そして、これらのプログラムは、フロッピーディスクやＣＤ－ＲＯＭ、あるいはその他のプログラム記録媒体に格納することができる。更に、通信回線等を利用してコンピュータに直接インストールして実施することが可能である。

【図面の簡単な説明】

【図１】本発明のフローチャート入力装置を示すブロック図である。

【図２】表示装置に表示されたウインドウの概略図である。

【図３】機能パーツの具体例説明図である。

【図４】ソフトウェア設計例説明図である。

【図５】フローチャート作成とプログラムへの変換例説明図である。

【図６】具体例２の装置ブロック図である。

【図７】挿入判定用表示例説明図（その１）である。

【図８】挿入判定用表示例説明図（その２）である。

【図９】具体例３の装置ブロック図である。

【図１０】挿入判定マトリクスの説明図である。

【図１１】具体例４の装置ブロック図である。

【図１２】接続判定動作の説明図（その１）である。

【図１３】接続判定動作の説明図（その２）である。

【図１４】誤ったフローチャート例の説明図である。

【図１５】具体例５の装置ブロック図である。

【図１６】変更前のフローチャート例説明図である。

【図１７】変更後のフローチャート例説明図である。

【図１８】フローチャートデータ例（変更前）の説明図である。

【図１９】フローチャートデータ例（変更後）の説明図である。

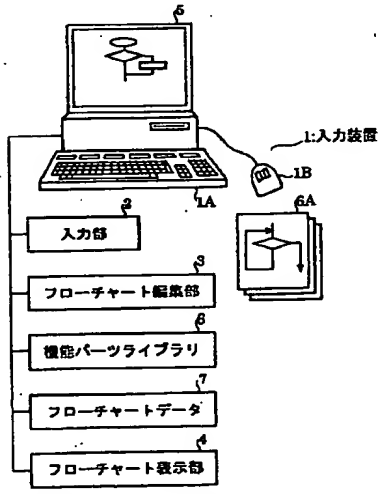
【図２０】具体例６の装置のブロック図である。

【図２１】状態遷移図の例説明図である。

【符号の説明】

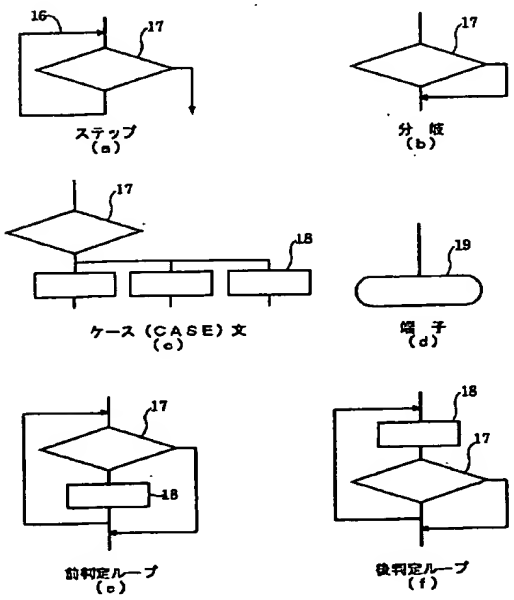
1　入力装置
2　入力部
3　フローチャート編集部
4　フローチャート表示部
5　表示装置
6　機能パーツライブラリ
6Ａ　機能パーツ
7　フローチャートデータ

【図１】



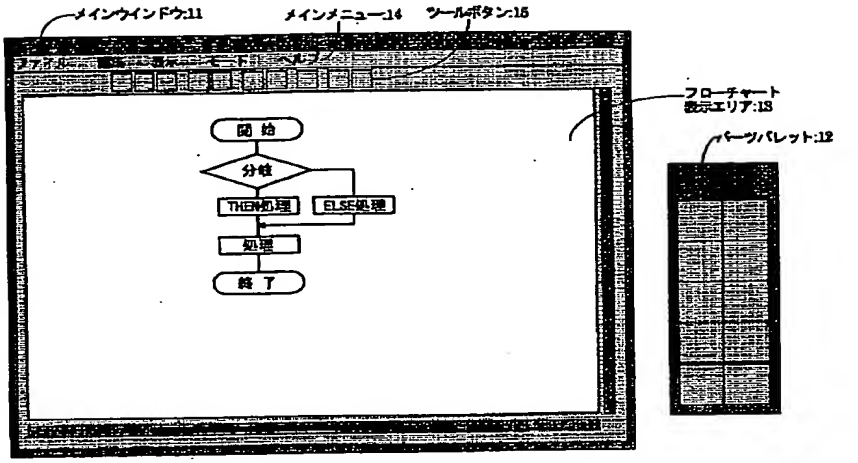
本発明のフローチャート入力装置ブロック図

【図３】



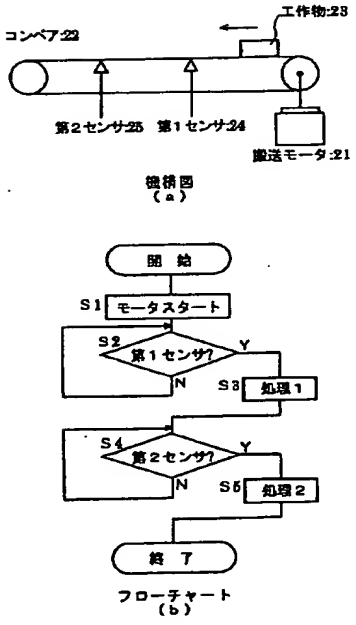
機能パーツの具体例説明図

【図２】



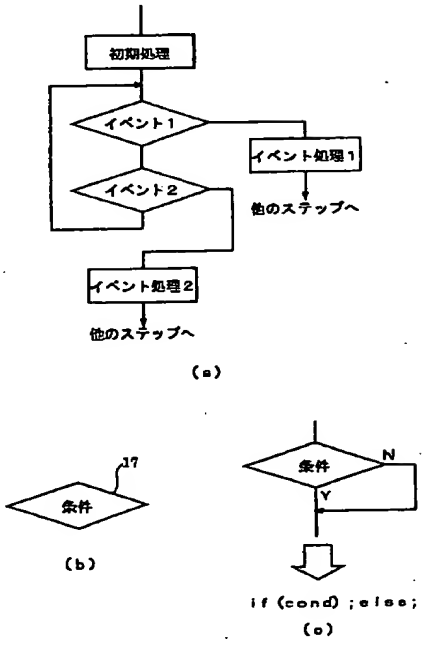
表示装置に表示されたウインドウの概略図

【図４】



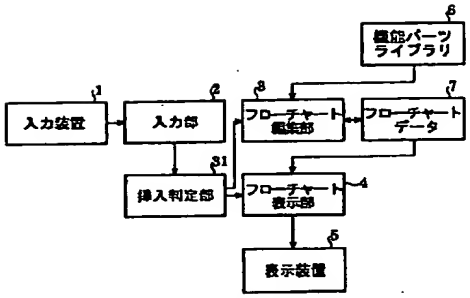
ソフトウェア設計例説明図

【図５】



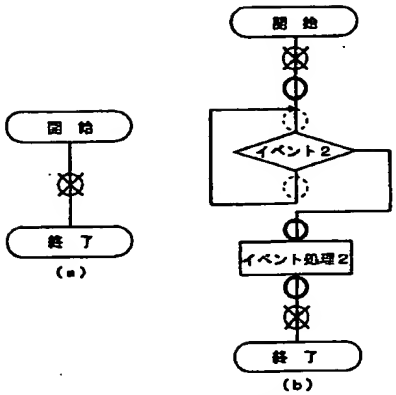
フローチャート作成とプログラムへの変換例

【図６】



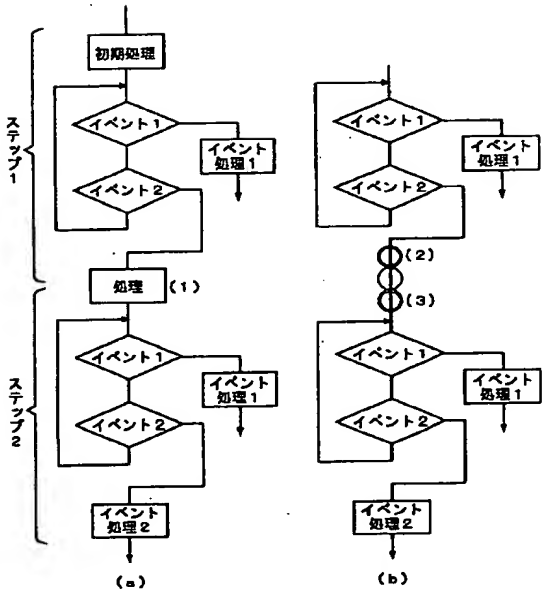
具体例２の装置ブロック図

【図７】



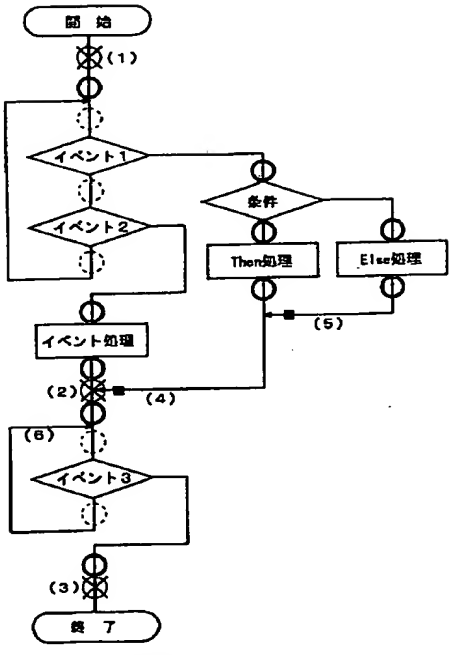
挿入判定用表示例説明図（その１）

【図８】



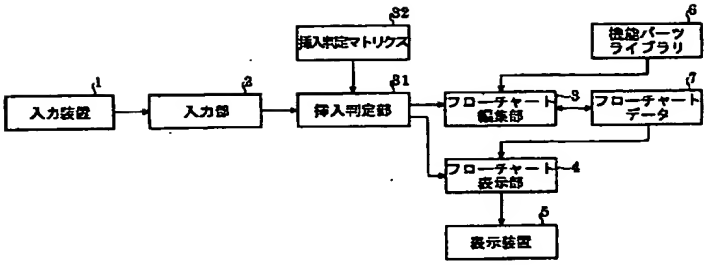
挿入判定用表示例説明図（その２）

【図１２】



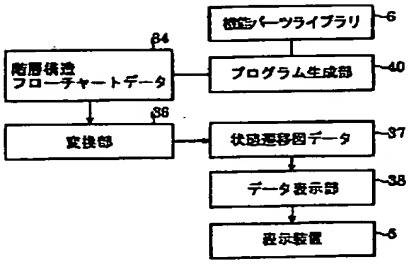
接続判定動作の説明図（その１）

【図９】



具体例３の装置ブロック図

【図２０】

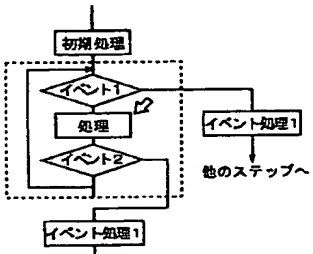


具体例６の装置のブロック図

【図１０】

	Step	分岐	イベント	処理	端子	前判定	後判定	ケース	結合子
○	×	○	×	○	×	○	○	○	×
●	○	×	×	×	○	×	×	×	○
破線○	×	×	○	×	×	×	×	×	×
○	×	○	×	×	×	×	×	×	×

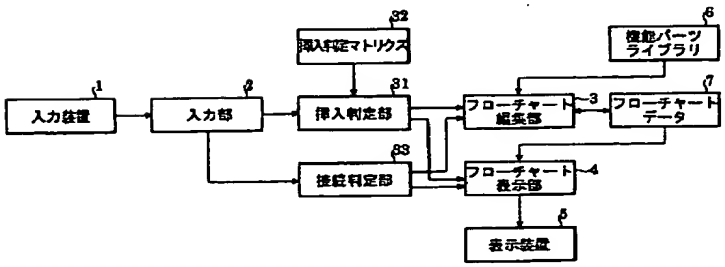
（a）



（b）

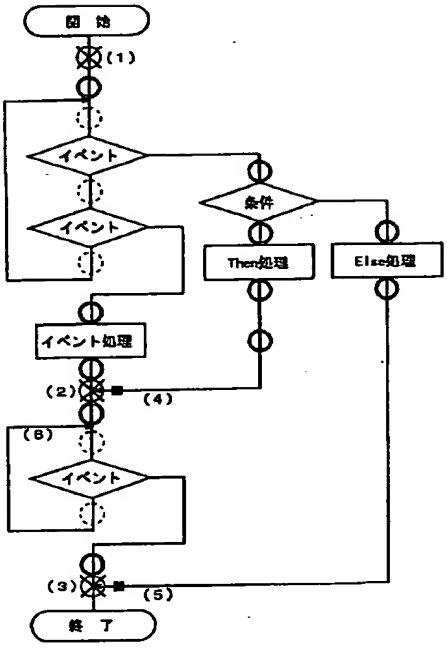
挿入判定マトリクスの説明図

【図１１】



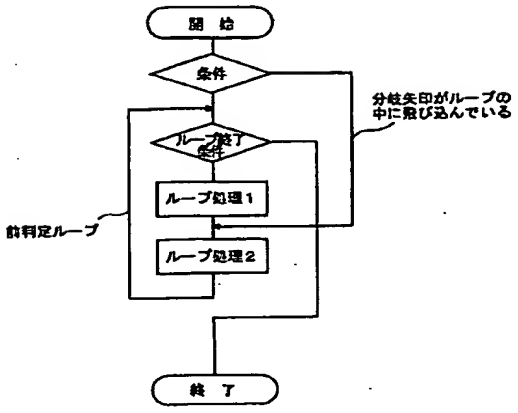
具体例４の装置ブロック図

【図１３】



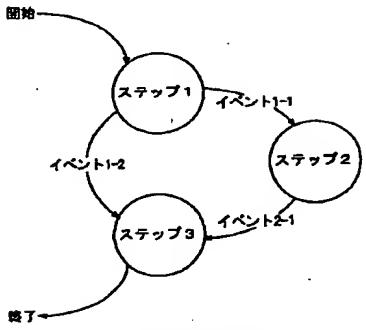
接続判定動作の説明図（その２）

【図１４】



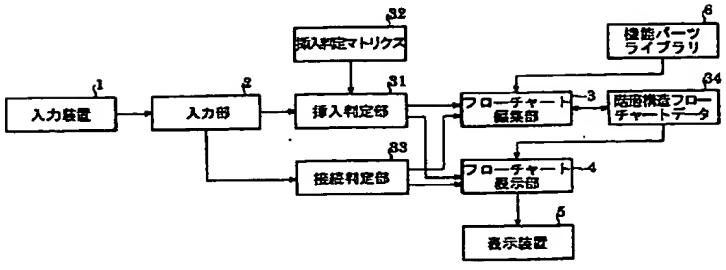
誤ったフローチャート例

【図２１】



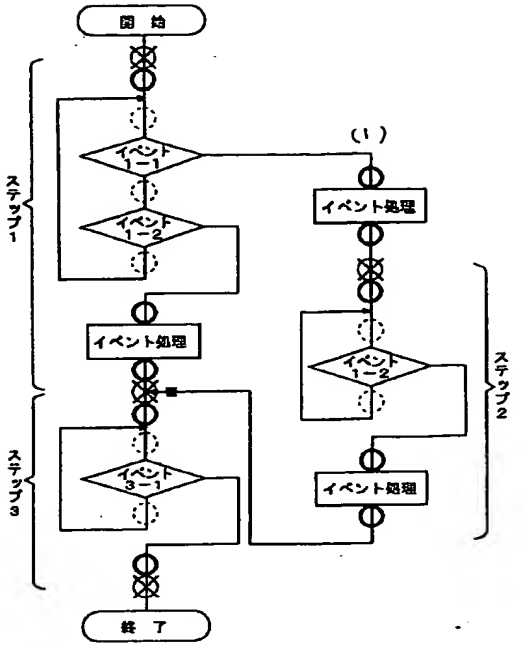
状態遷移図の例説明図

【図１５】



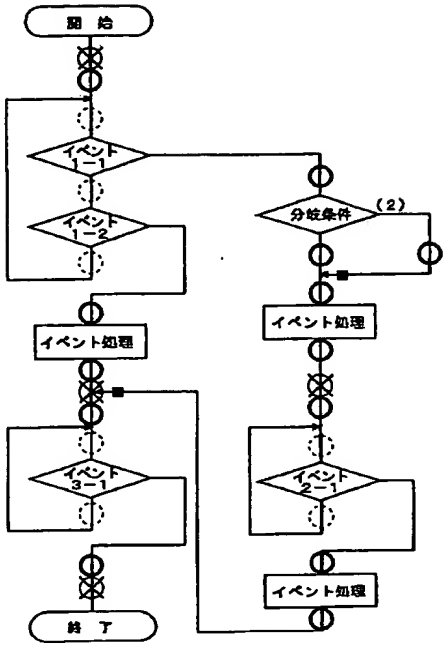
具体例５の装置ブロック図

【図１６】



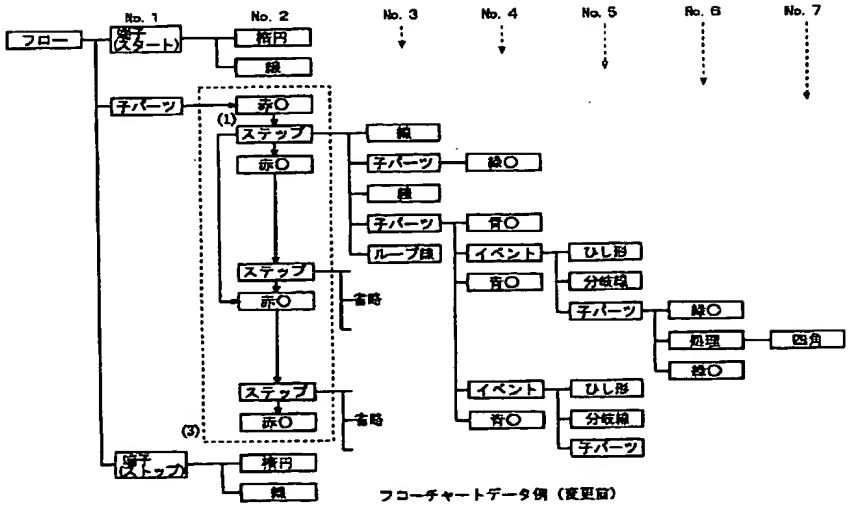
変更前のフローチャート例

【図１７】



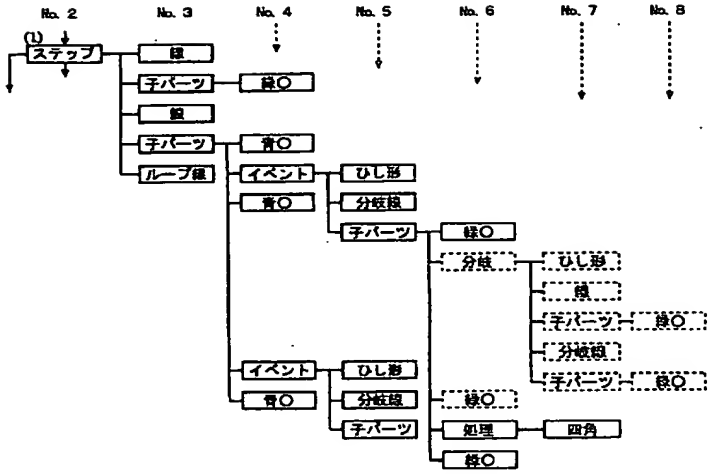
変更後のフローチャート例

【図１８】



フローチャートデータ例（変更前）

【図１９】



フローチャートデータ例（変更後）